Amendments to the Claims:

The following listing of the claims will replace all prior versions and listings of claims in the application.

Listing of the Claims:

Claims 1-5 (canceled)

6. (new) A computer implemented method of detecting vulnerabilities in a pre-existing source code listing, said source code listing having a listed sequence of expressions, each expression including a set of operands and operators to transform values of the operands, said listed sequence of expressions having an inherent control flow indicative of the run-time execution of the expressions and an inherent data flow indicative of the run-time transformations of operand values, said source code listing further having routine calls, said routine calls including arguments with which to invoke a routine, said arguments including expression-references and operand-references to computer files, said source code listing being stored in a computer-readable medium, said computer implemented method comprising the acts of:

executing computer instructions to analyze the source code listing to create computer models of said control flow to indicate the run-time sequence in which routine calls will be invoked and to create computer models of said arguments for the routine calls;

executing computer instructions to use said computer models of said control flow in order to determine a run-time sequence of execution of a pair of routine calls, said pair of routine calls having a first routine call and second routine call in

which execution of the first routine call precedes execution of said second routine call;

executing computer instructions to determine if a second routine to be executed has an argument referring to a file that is also referred to by an argument of the first routine to be executed and if so to identify said sequence as a race condition vulnerability;

generating a report that is viewable by a user and that identifies the race condition vulnerabilities, so the user may modify the source code listing to address the vulnerability if desired.

7. (new) The method of claim 6 further including the act of executing computer instructions to analyze the source code listing to create computer models of said data flow to indicate the run-time transformations of operand values and including the act of using data flow models to resolve the expression-references and operand-references to computer files in the first and second routine calls to detect whether both routines refer to the same computer file.

8. (new) The method of claim 6 wherein the control flow model is a control flow graph and wherein the method traverses the control flow graph backwards in order to determine the sequential relationship among routine calls in the source code listing.

9. (new) A system for detecting vulnerabilities in a pre-existing source code listing, said source code listing having a listed sequence of expressions, each expression including a set of operands and operators to transform values of the operands, said listed sequence of expressions having an inherent control flow indicative of the run-time execution of the expressions and an

inherent data flow indicative of the run-time transformations of operand values, said source code listing further having routine calls, said routine calls including arguments with which to invoke a routine, said arguments including expression-references and operand-references to computer files, said source code listing being stored in a computer-readable medium, said system comprising:

computer-executable instructions on a computer-readable medium to analyze the source code listing to create computer models of said control flow to indicate the run-time sequence in which routine calls will be invoked and to create computer models of said arguments for the routine calls;

computer-executable instructions on a computer-readable medium to use said computer models of said control flow to determine a run-time sequence of execution of a pair of routine calls, said pair of routine calls having a first routine call and second routine call in which execution of the first routine call precedes execution of said second routine call;

computer-executable instructions on a computer-readable medium to determine if the second routine to be executed has an argument referring to a file that is also referred to by an argument of the first routine to be executed and if so to identify said sequence as a race condition vulnerability;

computer-executable instructions on a computer-readable medium to generate a report that is viewable by a user and that identifies the race condition vulnerabilities, so the user may modify the source code listing to address the vulnerability if desired.

10. (new) The system of claim 9 further including computer-executable instructions on a computer-readable medium to analyze the source code listing to create computer models of said data flow to indicate the run-time transformations of operand values and to use the data flow models to resolve the expression-references and operand-references to computer files in the first and second routine calls to detect whether both routines refer to the same computer file.

11. (new) The system of claim 9 wherein the control flow model is a control flow graph and wherein the computer-executable instructions to use said computer models of said control flow to determine a run-time sequence of execution of a pair of routine calls, includes instructions to traverse the control flow graph backwards in order to determine the sequential relationship among routine calls in the source code listing.

Amendments to the Drawings:

The attached sheets of drawings include changes to Figures 1, 12 and 13. These sheets replace the original sheets to include the previously omitted reference signs.

Attachments: 3 Replacement Sheets (Figs. 1, 12 and 13)
3 Annotated Sheets Showing Changes (Figs. 1, 12 and 13)